

№ 2206

С.Ю. Муратова

Макросы и приложения

Курс лекций

№ 2206

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

Кафедра автоматизированных систем управления

С.Ю. Муратова

Макросы и приложения

Курс лекций

Рекомендовано редакционно-издательским
советом университета



Москва 2012

УДК 681.3
М91

Рецензент
канд. техн. наук, доц. *А.И. Широков*

Муратова, С.Ю.

М91 Макросы и приложения : курс лекций / С.Ю. Муратова. –
М. : Изд. Дом МИСиС, 2012. – 156 с.
ISBN 978-5-87623-623-4

Настоящий курс лекций разработан для семестрового модуля «Макросы и приложения» и посвящен разработке приложений на языке визуального программирования Visual Basic for Application в среде MS Excel.

Предназначен для бакалавров направлений 220700, 230100, 230400, 230700 и 231300.

УДК 681.3

ISBN 978-5-87623-623-4

© С.Ю. Муратова, 2012

ОГЛАВЛЕНИЕ

Введение.....	7
1. Объекты MS Excel.....	8
1.1. Объекты, их свойства и методы.....	8
1.1.1. Свойства: присвоение и использование значений.....	8
1.1.2. Примеры использования методов рабочей книги Excel.....	10
1.2. Ссылки на одиночные объекты и объекты из семейств.....	12
1.2.1. Различия между одиночными объектами и объектами из семейств.....	13
1.2.2. Семейства как объекты.....	14
1.2.3. Ссылка на объект.....	15
1.3. Иерархия объектов MS Excel.....	16
1.3.1. Иерархическая структура.....	16
1.3.2. Доступ к объектам через свойства и методы.....	20
1.4. Объект Application.....	21
1.4.1. Свойства объекта Application.....	21
1.4.2. Методы объекта Application.....	23
1.5. Объект Workbook и семейство Workbooks.....	23
1.5.1. Свойства объекта Workbook и семейства Workbooks.....	23
1.5.2. Методы объекта Workbook и семейства Workbooks.....	24
1.5.3. Функции, используемые для работы с файлами и папками.....	25
1.5.4. Функция MsgBox.....	28
1.5.5. Функция InputBox.....	32
1.6. Объект Worksheet.....	33
1.6.1. Свойства объекта Worksheet и семейства Worksheets.....	33
1.6.2. Методы объекта Worksheet и семейства Worksheets.....	33
1.7. Объект Range.....	34
1.7.1. Свойства объекта Range.....	35
1.7.2. Методы объекта Range.....	36
2. Использование переменных в VBA.....	38
2.1. Допустимые имена.....	38
2.2. Типы данных переменных VBA.....	38
2.3. Описание переменной.....	39
2.4. Использование переменных.....	40

2.5. Преимущества переменных.....	41
2.6. Объектные переменные	43
2.6.1. Задание объектной переменной.....	43
2.6.2. Объектные переменные общего типа	43
2.6.3. Объектные переменные конкретных типов.....	44
2.6.4. Преимущества объектных перемен.....	45
2.7. Неявное описание переменных и тип Variant.....	46
2.8. Обязательное описание переменных.....	48
2.9. Типы данных по умолчанию	50
2.10. Пользовательские типы данных	50
3. Массивы VBA	53
3.1. Размерность массива.....	53
3.2. Объявление массива.....	54
3.3. Использование массива	55
3.4. Номер первого элемента и границы массива.....	58
3.5. Динамические массивы	59
3.6. Сохранение данных в динамическом массиве при изменении последней размерности	61
3.7. Пять функций для работы с массивами	62
4. Константы	66
5. Вызов одной программы из другой	67
5.1. Фрагментирование кода	67
5.2. Передача данных при вызове программы.....	68
6. Использование функций в VBA.....	72
7. Область видимости переменных, констант, подпрограмм и функций	74
7.1. Область видимости переменных.....	74
7.1.1. Переменные уровня процедуры	75
7.1.2. Переменные уровня модуля.....	76
7.1.3. Переменные уровня проекта.....	77
7.1.4. Сохраняемые переменные.....	79
7.1.5. Область видимости подпрограмм и функций	80
7.1.6. Сохраняемые подпрограммы и функции.....	81
8. Управляющие структуры.....	82
8.1. Управляющая инструкция If-Then-Else.....	82
8.2. Управляющая инструкция Select Case	86
8.3. Управляющая инструкция For-Next	89
8.4. Управляющая инструкция For-Each-Next.....	91
8.4.1. Инструкция For-Each-Next с многомерными массивами	93

8.4.2. Инструкция For-Each-Next с семействами	94
8.5. Управляющая инструкция While-Wend	95
8.6. Управляющая инструкция Do-Loop	96
9. Инструкция With.....	99
10. Встроенные функции VBA	101
10.1. Математические функции	101
10.2. Функции проверки типов	101
10.3. Функции преобразования форматов.....	102
10.4. Функции обработки строк	103
10.5. Функции времени и даты.....	105
11. Обработка ошибок.....	106
11.1. Предотвращение ошибок программными средствами	106
11.2. Обработка ошибок, инструкция On Error	108
12. Разработка пользовательского интерфейса.....	111
12.1. Форма (UserForm).....	111
12.1.1. Вставка формы	111
12.1.2. Основные свойства и методы формы.....	112
12.1.3. События формы.....	113
12.2. Элементы управления формы VBA.....	114
12.2.1. Некоторые общие свойства элементов управления	114
12.2.2. Соглашения об именах	116
12.2.3. Некоторые общие методы элементов управления.....	117
12.2.4. Общие события элементов управления	117
12.3. Кнопка (CommandButton)	119
12.4. Поле (TextBox).....	120
12.5. Надпись (Label).....	121
12.6. Список (ListBox).....	124
12.6.1. Основные свойства элемента управления ListBox.....	125
12.6.2. Методы ListBox	127
12.6.3. Заполнение списка	127
12.6.4. Пример создания списка	128
12.6.5. Определение выбранных элементов списка.....	131
12.7. Поле со списком (ComboBox)	132
12.8. Флажок (CheckBox).....	134
12.9. Выключатель (ToggleButton).....	136
12.10. Переключатель (OptionButton).....	137
12.11. Полоса прокрутки (ScrollBar) и счетчик (SpinButton)	138

12.12. Создание нестандартных меню и панелей инструментов	142
12.12.1. Объект CommandBar и семейство CommandBars	142
12.12.2. Методы объекта CommandBar	143
12.12.3. Свойства объекта CommandBar	144
12.12.4. Семейство CommandBarControls и объект CommandBarControl	145
12.12.5. Пример создания/удаления панели инструментов	147
12.12.6. Пример создания/удаления меню	149
13. Обработка событий объектов Workbook и Worksheet	151
13.1. События объекта Workbook	151
13.1.1. Событие Open	151
13.1.2. Событие BeforeClose	152
13.1.3. Событие SheetActivate	152
13.2. События объекта Worksheet	152
13.2.1. Событие Activate	153
13.2.2. Событие Deactivate	154
13.2.3. Событие SelectionChange	154
Библиография	155

ВВЕДЕНИЕ

Одним из эффективных средств создания информационных систем или автоматизированных рабочих мест является программа Microsoft Excel, которая предоставляет разработчику возможность использовать одновременно преимущества визуального программирования и электронной таблицы.

Языком визуального программирования в Microsoft Excel, равно как в других приложениях Microsoft Office, является Visual Basic for Applications (VBA). VBA можно отнести к языкам объектно-ориентированного программирования (ООП), в которых данные и код объединяются в нечто единое целое, называемое объектом.

Использование VBA актуально при разработке специфических приложений в малобюджетных организациях независимо от рода их деятельности, особенно если циркулирующая в них информация хранится в виде таблиц или баз данных Excel.

1. ОБЪЕКТЫ MS EXCEL

Основным понятием в MS Excel является объект. Говоря коротко, *объект – это нечто, чем можно управлять и что можно программировать*. Модель объектов Excel содержит более 100 собственных элементов и несколько – общих для всех приложений Office. Диапазон объектов Excel очень широк – от простых прямоугольников или текстовых полей до таких сложных структур, как сводные таблицы и диаграммы.

Каждый из объектов Excel предназначен для выполнения определенного действия, необходимого для анализа данных. Создание приложения заключается в объединении нужных объектов средствами языка программирования VBA.

1.1. Объекты, их свойства и методы

Каждый объект Excel располагает набором *свойств* (properties) и *методов* (methods).

Можно сказать, что свойства – это прилагательные, описывающие объект, а методы – глаголы, означающие действия, которые могут быть выполнены самим объектом или над ним.

1.1.1. Свойства: присвоение и использование значений

1.1.1.1. Присвоение значений

Рассмотрим в качестве примера объекта рабочую книгу (**Workbook**) – документ Excel. Вот некоторые свойства этого объекта¹:

Свойство	Описание
Author	Имя пользователя, создавшего рабочую книгу
HasPassword	True , если рабочая книга защищена паролем, и False – в противном случае
Name	Название рабочей книги
Path	Путь к файлу книги на диске
ReadOnly	True , если сохранение рабочей книги запрещено, и False – в противном случае

¹ Полный перечень свойств можно просмотреть с помощью Object Browser.

Над свойством можно выполнять две операции: задать его значение или использовать его. И в том, и в другом случае необходимо указать имя объекта и имя свойства, разделив их точкой. Для определения значения свойства используется знак равенства (=). Например, инструкция для присваивания значения свойству **Author** рабочей книги **Мои таблицы.xls** выглядит так:

```
Workbooks ("Мои таблицы.xls").Author = "Муратова С.Ю."
```

Структура этой строки такова:

Workbooks ("Мои таблицы.xls")	Имя объекта;
.	Точка (разделитель);
Author	Имя свойства;
=	Знак присваивания;
"Муратова С.Ю."	Значение свойства.

ВНИМАНИЕ! Чтобы данная инструкция превратилась в подпрограмму VBA, перед ней нужно вставить строку со словом **Sub** и названием подпрограммы, а после нее – строку со словами **End Sub**, как показано ниже:

```
Sub ЗадатьАвтора ()
    Dim Автор As String
    Workbooks ("Мои таблицы.xls").Author _
    = "Муратова С.Ю."
End Sub
```

По мере знакомства со свойствами рабочей книги мы узнаем и несколько важных правил.

Во-первых, с каждым свойством связано значение, которое должно согласовываться с его типом (например, имя рабочей книги – строковое, свойство **HasPassword** – логическое и т.д.).

Во-вторых, свойство может принадлежать одному или нескольким различным объектам. Например, в Excel свойство **HasPassword** есть только у объекта **Workbook**, а вот свойство **Name** – практически у всех объектов Excel.

В-третьих, есть свойства, которые можно только использовать¹, но нельзя изменять. Таковым, например, является свойство **Path**.

¹Так называемые свойства только для чтения (read-only).

1.1.1.2. Использование значений

Для использования значения свойства применяют ту же инструкцию VBA, что и для присвоения, но ее элементы располагают в обратном порядке. Чтобы «извлечь» значение свойства, его обычно присваивают какой-либо переменной. В строке программы, приведенной ниже, значение свойства **Author** объекта **Workbook** присваивается строковой переменной Автор.

```
Sub УзнатьАвтора ()  
    Dim Автор As String  
    Автор = Workbooks("Мои таблицы.xls").Author  
    MsgBox "Автор этой книги: " & Автор  
End Sub
```

В данном фрагменте программы:

1) оператор **Dim** объявляет переменную Автор как строковую переменную;

2) оператор конкатенации **&** объединяет содержимое переменной Автор со строкой "Автор этой книги: " и отправляет полученное сообщение во встроенную функцию **MsgBox**, которая выводит его на экран.

1.1.2. Примеры использования методов рабочей книги Excel

Объект **Workbook** также содержит множество методов, определяющих действия, совершаемые им или над ним. Ниже приведены некоторые из них.

Метод	Действие
Activate	Активизация первого окна, связанного с книгой
Close	Закрытие книги
PrintPreview	Предварительный просмотр книги перед печатью
Protect	Защита книги паролем
Save	Сохранение книги

1.1.2.1. Вызов метода

Синтаксис команды VBA для вызова метода отличается от синтаксиса команды присвоения значения свойству. Все, что нужно в этом случае, – это указать объект и метод. Кроме того, в

большинстве методов Excel используются *аргументы*, или *параметры*, – дополнительные данные для управления способом выполнения метода. Некоторые из аргументов необязательные, т.е. в зависимости от потребностей вы можете указывать все, некоторые или ни одного из них. Например, метод **Close** объекта **Workbook** имеет три необязательных аргумента:

Свойство	Значение
saveChanges	Принимает значения True (сохранить изменения в файле) или False (не сохранять изменения)
fileName	Имя файла для сохранения книги, если предыдущий аргумент имеет значение True
routeWorkbook	Принимает значения True (отправить книгу по маршруту) или False (не отправлять книгу)

При вызове метода без аргументов им присваиваются значения, заданные по умолчанию. Например, при отсутствии первого аргумента **saveChanges** в методе **Close** ему будет присвоено значение **True**. Аргумент **fileName** по умолчанию содержит текущее имя файла, а аргумент **routeWorkbook** – значение **False**. Вызов метода без аргументов запишется следующим образом:

```
Workbooks("Мои таблицы.xls").Close
```

1.1.2.2. Передача аргумента в метод

Существуют два способа передачи аргументов в метод: по позиции и по имени.

При передаче аргументов *по позиции* вы просто добавляете их к вызову метода, разделяя запятыми. ОБРАТИТЕ ВНИМАНИЕ на порядок следования аргументов! Для метода **Close**, например, правильный порядок таков: **saveChanges**, **fileName**, **routeWorkbook**.

Ниже приведен пример вызова метода **Close** со всеми тремя аргументами, переданными по позиции:

```
Workbooks("Мои таблицы.xls").Close True, _
    "Мои таблицы-2.xls", False
```

В данном примере метод **Close** закрывает рабочую книгу Мои таблицы.xls, сохраняя её под именем Мои таблицы-2.xls.

ОБРАТИТЕ ВНИМАНИЕ на символ подчеркивания в конце первой строки. Он означает, что команда продолжается на следующей строке. Перед символом подчеркивания необходимо ввести пробел!

Чтобы пропустить какой-то аргумент (например, второй), вставьте вместо него пробел:

```
Workbooks("Мои таблицы.xls").Close True, , False
```

При передаче аргумента *по имени* надо указать в вызове три элемента:

- имя аргумента (например, **saveChanges**);
- оператор присваивания с двоеточием (**:=**);
- значение аргумента.

В следующем примере метод **Close** вызывается с аргументами, передаваемыми по имени:

```
Workbooks("Мои таблицы.xls").Close _  
saveChanges:=True, fileName:="Мои таблицы-2.xls", _  
routeWorkbook:=False
```

Передавая аргументы по имени, не обязательно соблюдать их порядок. Приведенный ниже код идентичен предыдущему:

```
Workbooks("Мои таблицы.xls").Close _  
routeWorkbook:=False, savChanges:=True, _  
fileName:="Мои таблицы-2.xls"
```

При передаче аргументов по имени программа становится более понятной, а по позиции – более короткой.

1.2. Ссылки на одиночные объекты и объекты из семейств

Как уже говорилось, при обращении к свойству или методу объекта их имена просто добавляют к имени объекта, отделяя от него точкой. Ссылка же на сам объект может выглядеть по-разному. К одиночному объекту *семейства* обращаются либо по имени, либо по его номеру в *семействе* объектов. Чтобы разобраться, чем отличаются два этих способа, уясним себе, что такое *семейство* (collection). Если не вдаваться в детали, *семейство – это группа похожих объектов*. Все объекты Excel можно отнести к одному из двух приблизительно равных по численности классов: *одиночным*

*объектам и объектам из семейства*¹. К первым обращаются непосредственно, ко вторым – по имени или по номеру. В предыдущих примерах обращение к рабочей книге Мои_таблицы.xls было оформлено с помощью указателя на объект семейства **Workbooks**² по имени.

1.2.1. Различия между одиночными объектами и объектами из семейств

Как же узнать, является объект одиночным или входит в семейство? Для этого надо применить простые, интуитивно понятные правила.

Правило 1. Одиночный объект может существовать в данный момент времени только в одном экземпляре. Другими словами, он уникален.

Правило 2. Объекты из семейства могут существовать одновременно в нескольких экземплярах.

В Excel может быть открыто несколько рабочих книг, поэтому к объекту Мои_таблицы.xls можно было бы, согласно правилу 2, обратиться по индексу, предварительно узнав его порядковый номер. Следующий пример демонстрирует эту возможность:

```
Sub СсылкаНаКнигуПоИндексу ()
```

```
'1. Объявляем переменные:
```

```
    Dim Автор As String  
    Dim КолОткрытыхКниг As Integer
```

```
'2. Определяем количество открытых книг:
```

```
    КолОткрытыхКниг = Workbooks.Count
```

```
'3. Ищем среди открытых книг книгу с именем Мои_таблицы.xls:
```

```
For i = 1 To КолОткрытыхКниг  
    If Workbooks(i).Name = "Мои_таблицы.xls" Then  
        Автор = Workbooks(i).Author
```

¹ Из этой классификации слегка выбиваются три объекта Excel: Range, Sheets и Shapes. Объект Range считается одиночным, но обладает также характеристиками семейства. Sheets и Shapes — семейства, но не объектов, а других семейств.

² Семейство рабочих книг

```

MsgBox "Автор этой книги: " & Автор
Exit Sub
End If
Next i
MsgBox "Книга Мои таблицы.xls среди открытых книг не найдена!"
End Sub

```

В этом примере обратите внимание на комментарии – текст, расположенный после апострофа ('). Программу с пояснительными комментариями значительно легче отлаживать и вносить в неё изменения. Помните, что в программе без комментариев её детали забываются через удивительно короткое время!

ЗАМЕЧАНИЯ. 1. Комментарии должны содержать некоторую дополнительную информацию, а не перефразировать программу. С этой точки зрения 1-й и 2-й комментарии правомерны как пояснения только в учебной программе.

2. Обязательно делайте вводные комментарии. Они должны включать в себя следующие пункты:

- назначение программы;
- указания по вызову программы и её использованию;
- список и назначение основных переменных и массивов;
- указания по вводу-выводу, список всех файлов;
- список используемых подпрограмм и их назначение;
- требования к компьютеру;
- сведения об авторе;
- дату написания программы.

3. Для улучшения наглядности программы вставляйте пустые строки и делайте отступы.

1.2.2. Семейства как объекты

Хотя семейство объединяет несколько объектов, оно и само является объектом, причем одиночным. Концепцию семейства лучше проиллюстрировать на конкретном примере.

Семейство **Workbooks** содержит несколько объектов **Workbook**, с каждым из которых связан набор свойств и методов. Другой набор свойств и методов связан с семейством **Workbooks**. Таким образом, в Excel есть три вида объектов: *одиночный объект, объект семей-*